Concurrency in Java

Concurrency allows to partition a program into separate independently running tasks.
Each task is driven by a thread of execution.
A thread is a single sequential flow of control within a process. A single process can have multiple concurrently executing tasks.
The CPU's time is being sliced among all the tasks.

[B Eckel, Thinking in Java, 4th Edition, Chapter: Concurrency]

Defining a Thread - I

```
public class MyThread1 extends Thread{
    private int counter;
    private String id;

    public MyThread1(int counter, String id) {
        this.counter = counter;
        this.id = id;
    }

    public void run() {
        for(int i = 0; i < counter; i++)
            System.out.println(id + i + " ");
    }
}
```

```
public class MyThread1Main {
    public static void main(String[] args) {
        MyThread1 mt1 = new MyThread1(2000, "TT");
        MyThread1 mt2 = new MyThread1(1000, "BB");
        mt1.start();
        mt2.start();
    }
}
```

```
TT669
BB466
BB467
TT670
BB468
TT671
BB469
TT672
BB470
TT673
BB471
TT674
BB472
TT675
BB473
TT676
BB474
TT677
BB475
TT678
BB476
TT679
BB477
```

a part of the output showing that the tasks are mixed together

Defining a Thread - 2

```
public class MyThread2 implements Runnable{
    private int counter;
    private String id;

    public MyThread2(int counter, String id) {
        this.counter = counter;
        this.id = id;
    }

    public void run() {
        for(int i = 0; i < counter; i++)
            System.out.println(id + i + " ");
    }
}


public class MyThread2Main {
    public static void main(String[] args) {
        Thread mt1 = new Thread(new MyThread2(2000, "TT"));
        Thread mt2 = new Thread(new MyThread2(1000, "BB"));
        mt1.start();
        mt2.start();
    }
}
```

the synchronized keyword

```
public class Square {
    private int width, height;

    public void set(int side) {
        this.width = side;
        this.height = side;

        if(width == height)
            System.out.println("Side " + width);
        else
            System.out.println("ERROR");
    }
}

public class MySquareMain {
    public static void main(String[] args) {
        Square s = new Square();

        MySquareThread t1 = new MySquareThread(s);
        MySquareThread t2 = new MySquareThread(s);
        MySquareThread t3 = new MySquareThread(s);
        MySquareThread t4 = new MySquareThread(s);

        t1.run();
        t2.run();
        t3.run();
        t4.run();
    }
}
```

```
class MySquareThread extends Thread{
    private Square square;

    public MySquareThread(Square square) {
        this.square = square;
    }

    public void run() {
        for(int i = 0; i < 200000; i++)
            square.set(i);
    }
}
```

the synchronized keyword

```
public class Square {
    private int width, height;

    public synchronized void set(int side) {
        this.width = side;
        this.height = side;

        if(width == height)
            System.out.println("Side " + width);
        else
            System.out.println("ERROR");
    }
}
```

ERROR will never be printed!!!

wait(), notify(), notifyAll()

```
public class Resource {
    private int element;
    private boolean isElement = false;

    public synchronized void put(int e) {
        while(isElement) {
            try { wait(); }
            catch(InterruptedException ex) {}
        }
        element = e;
        isElement = true;
        notifyAll();
    }

    public synchronized int get() {
        while(!isElement) {
            try { wait(); }
            catch(InterruptedException ex) {}
        }
        isElement = false;
        notifyAll();
        return element;
    }
}
```

wait(), notify(), notifyAll()

```
class GetThread extends Thread {
    private Resource r;

    public GetThread(Resource r) {
        this.r = r;
    }

    public void run() {
        while(true)
            System.out.println(r.get());
    }
}
class PutThread extends Thread {
    private Resource r;

    public PutThread(Resource r) {
        this.r = r;
    }

    public void run() {
        int i = 0;
        while(true) {
            i++;
            r.put(i % 100);
        }
    }
}
```

```
class ConsumerProducer {
    public static void main(String[] argv) {
        Resource r = new Resource();
        new GetThread(r).start();
        new PutThread(r).start();
    }
}
```